

IN THE CLAIMS

1. (Previously Presented) A method for circuit emulation over a multi-packet label switching (MPLS) network, comprising:

receiving a time division multiplexed data stream at an ingress end from an optical carrier;

dividing said data stream into a set of fixed sized packets;

adding a service header to each of said packets;

adding an additional header on top of said service header in accordance with MPLS protocols;

removing said additional header after each packet has been processed by said MPLS network; and

using said service header to recover said data stream at an egress end, wherein said service header includes a structure pointer to indicate whether a header byte indicating a start of a synchronous payload envelope is present within a packet, said structure pointer indicating a location of said header byte in said packet.

2. (Previously Presented) The method of claim 1, further comprising:

monitoring said data stream; and

attaching an alarm bit in a service header of a subsequent packet if a break in said data stream is detected.

3. (Currently Amended) The method of claim 1, further comprising:

~~using a structure pointer in said service header to indicate whether a header byte in a synchronous payload envelope is present within a packet, said structure pointer indicating the location of said header byte in said packet~~
negative justification bit and a positive justification bit in said service header to indicate whether said synchronous payload envelope includes a negative stuff byte or a positive stuff byte.

4. (Currently Amended) The method of ~~claim 3~~ claim 1, further comprising:

reserving a pointer value indicating that said header byte is not present within said packet.

5. (Previously Presented) The method of claim 1, further comprising:

recording a stuffing time difference in a service header at said ingress end; and

implementing said stuffing time difference at said egress end.

6. (Previously Presented) The method of claim 1, further comprising:

- storing a first set of frames into a data buffer;
- calculating a first data average of said first set of frames in said data buffer to obtain a threshold value;
- storing a next set of frames into said data buffer;
- calculating a next data average of said next set of frames in said data buffer;
- comparing said next data average to said threshold value;
- if said next data average is greater than said threshold value:

- generating a negative justification indicator; and
 - sending one more byte at said egress end;
- if said next data average is less than said threshold value:

- generating a positive justification indicator; and
 - sending one less byte at said egress end.

7. (Previously Presented) The method of claim 1, further comprising:

- checking a sequence counter in said service header of each packet in said set of packets;
- locating at least one header byte in said set of packets;
- measuring all bytes between two header bytes; and
- pushing said set of packets into a frame.

8. (Previously Presented) The method of claim 1, further comprising:

checking a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially; and

inserting a dummy packet if a packet is missing in said set of packets.

9. (Previously Presented) The method of claim 8, further comprising:

receiving an out of sequence packet; and
discarding said out of sequence packet.

10. (Previously Presented) The method of claim 1, further comprising:

checking a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially;

terminating a current connection if multiple packets are missing in said set of packets;

discarding said set of packets; and
establishing a new connection to begin receiving packets.

11. (Previously Presented) The method of claim 1, further comprising:

checking a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially; and

establishing an in-frame condition after said set of packets are received in sequence.

12. (Previously Presented) The method of claim 11, further comprising:

determining whether said in-frame condition is valid; and
terminating a current connection if said in-frame condition is not valid.

13. (Previously Presented) A computer readable medium including code for circuit emulation over a multi-packet label switching (MPLS) network, the code operable to:

receive a time division multiplexed data stream at an ingress end from an optical carrier;

divide said data stream into a set of fixed sized packets;

add a service header to each of said packets;

add an additional header on top of said service header in accordance with MPLS protocols;

remove said additional header after each packet has been processed by said MPLS network; and

use said service header to recover said data stream at an egress end, wherein said service header includes a structure pointer to indicate whether a header byte indicating a start of a synchronous payload envelope is present within a packet, said structure pointer indicating a location of said header byte in said packet.

14. (Previously Presented) The computer readable medium of claim 13, wherein the code is further operable to:

monitor said data stream; and

attach an alarm bit in a service header of a subsequent packet if a break in said data stream is detected.

15. (Currently Amended) The computer readable medium of claim 13, wherein the code is further operable to:

~~use a structure pointer in said service header to indicate whether a header byte in a synchronous payload envelope is present within a packet, said structure pointer indicating the location of said header byte in said packet~~
negative justification bit and a positive justification bit in said service header to indicate whether said synchronous payload envelope includes a negative stuff byte or a positive stuff byte.

16. (Currently Amended) The computer readable medium of ~~claim 15~~ claim 13, wherein the code is further operable to:

reserve a pointer value indicating that said header byte is not present within said packet.

17. (Previously Presented) The computer readable medium of claim 13, wherein the code is further operable to:

record a stuffing time difference in a service header at said ingress end; and

implement said stuffing time difference at said egress end.

18. (Previously Presented) The computer readable medium of claim 13, wherein the code is further operable to:

- store a first set of frames into a data buffer;
- calculate a first data average of said first set of frames in said data buffer to obtain a threshold value;
- store a next set of frames into said data buffer;
- calculate a next data average of said next set of frames in said data buffer;
- compare said next data average to said threshold value;
- if said next data average is greater than said threshold value:

- generate a negative justification indicator; and
 - send one more byte at said egress end;
- if said next data average is less than said threshold value:

- generate a positive justification indicator; and
 - send one less byte at said egress end.

19. (Previously Presented) The computer readable medium of claim 13, wherein the code is further operable to:

- check a sequence counter in said service header of each packet in said set of packets;
- locate at least one header byte in said set of packets;
- measure all bytes between two header bytes; and
- push said set of packets into a frame.

20. (Previously Presented) The computer readable medium of claim 13, wherein the code is further operable to:

check a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially; and

insert a dummy packet if a packet is missing in said set of packets.

21. (Previously Presented) The computer readable medium of claim 20, wherein the code is further operable to:

receive an out of sequence packet; and

logic code for discarding said out of sequence packet.

22. (Previously Presented) The computer readable medium of claim 13, wherein the code is further operable to:

check a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially;

establish an in-frame condition after all packets for a frame are received in sequence;

terminate a current connection if multiple packets are missing in said set of packets;

discard said set of packets; and

establish a new connection to begin receiving packets.

23. (Previously Presented) The computer readable medium of claim 22, wherein the code is further operable to:

check a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially; and

establish an in-frame condition after the set of packets are received in sequence.

24. (Previously Presented) The computer readable medium of claim 23, wherein the code is further operable to:

determine whether said in-frame condition is valid; and

terminate a current connection if said in-frame condition is not valid.